Abstract

Simulation models and simulation experiments are increasingly complex. One way to handle this complexity is developing software languages tailored to specific application domains, so-called domain-specific languages (DSLs). This thesis explores the potential of employing DSLs in modeling and simulation. We study different DSL design and implementation techniques and illustrate their benefits for expressing simulation models as well as simulation experiments with several examples.

Regarding simulation models, we focus on discrete-event models based on continuous-time Markov chains (CTMCs). Most of our work revolves around ML-Rules, an rule-based modeling language for biochemical reaction networks. First, we relate the expressive power of ML-Rules to other currently available modeling languages for this application domain. Then we define the abstract syntax and operational semantics for ML-Rules, mapping models to CTMCs in an unambiguous and precise way. Based on the formal definitions, we present two approaches to implement ML-Rules as a DSL. The core of both implementations is finding the matches for the patterns on the left side of ML-Rules' rules. The first approach makes use of ideas from functional programming and realizes ML-Rules as an internal DSL embedded in Scala. The second approach utilizes the language workbench Xtext and object-oriented programming to implement ML-Rules as an external DSL. We relate both approaches to each other and discuss their specific trade-offs, for example regarding computational efficiency. In addition, we demonstrate how DSL implementation techniques can be employed to integrate CTMC-based modeling into Repast Simphony, a software framework for agent-based simulation.

Simulation experiments benefit from DSLs in a different way than simulation models. Here, we focus on more technical issues like repeatability, replicability, reproducibility, and reusability of experiments. The utility of DSLs for expressing simulation experiments is illustrated based on two different DSL implementations. First, we discuss SESSL, an object-oriented Scala-based internal DSL, highlighting diverse ways in which DSLs can support effective simulation experimentation. Based on the ideas of SESSL, we also present a Scala-based internal DSL that is implemented in a purely functional fashion, and discuss the implications.

Zusammenfassung

Simulationsmodelle und -experimente werden immer komplexer. Eine Möglichkeit, dieser Komplexität zu begegnen, ist, auf bestimmte Anwendungsgebiete spezialisierte Softwaresprachen, sogenannte domänenspezifische Sprachen (*DSLs, domain-specific languages*), zu entwickeln. Die vorliegende Arbeit untersucht, wie DSLs in der Modellierung und Simulation eingesetzt werden können. Wir betrachten verschiedene Techniken für Entwicklung und Implementierung von DSLs und illustrieren ihren Nutzen für das Ausdrücken von Simulationsmodellen und -experimenten anhand einiger Beispiele.

In Bezug auf Simulationsmodelle konzentrieren wir uns auf diskret-ereignisbasiert Modelle, die auf Markow-Ketten mit kontinuierlicher Zeitbasis (*CTMCs, continuous-time Markov chains*) basieren. Der größte Teil dieses Abschnitts dreht sich um die regelbasierte Modellierungssprache für biochemische Reaktionsnetze ML-Rules. Als Ausgangspunkt setzen wir die Ausdruckskraft von ML-Rules mit anderen verfügbaren Modellierungssprachen in diesem Anwendungsgebiet in Beziehung. Im Anschluss definieren wir die abstrakte Syntax und formale Semantik von ML-Rules, wodurch eine eindeutige Abbildung von Modellen auf CTMCs hergestellt wird. Auf Grundlage dieser formalen Definitionen stellen wir zwei Ansätze zur Implementierung von ML-Rules in einer DSL vor. Der Kern beider Implementierungen ist das Finden der Vorkommen der Muster auf den linken Regelseiten von ML-Rules. Der erste Ansatz nutzt Ideen aus der funktionalen Programmierung und setzt ML-Rules als interne DSL in Scala um. Der zweite Ansatz nutzt die *language workbench* Xtext und objekt-orientiere Programmierung, um ML-Rules als externe DSL umzusetzen. Wir stellen beide Ansätze nebeneinander und diskutieren die spezifischen Trade-offs, zum Beispiel in Bezug auf Recheneffizienz. Des Weiteren demonstrieren wir, wie CTMC-basierte Modellierung mit Hilfe von DSL-Implementierungstechniken in Repast Simphony, ein Framework für agentenbasierte Simulation, integriert werden kann.

Simulationsexperiment profitieren auf andere Weise als Simulationsmodelle von der Nutzung von DSLs. Hier konzentrieren wir uns auf technischere Aspekte wie Wiederholbarkeit, Replizierbarkeit, Reproduzierbarkeit und Wiederverwendbarkeit von Experimenten. Der Nutzen von DSLs für das Ausdrücken von Simulationsexperimenten wird mit Hilfe zweier DSl-Implementierungen gezeigt. Zunächst behandeln wir SESSL, eine objektorientierte Scala-basierte interne DSL, wobei wir hervorheben, wie die Nutzung von DSLs das effektive Durchführen von Simulationsexperimenten unterstützt. Aufbauend auf den Ideen hinter SESSL stellen wir dann eine Scala-basierte interne DSL vor, die auf rein funktionale Art implementiert ist, und diskutieren die Implikationen.